Minimum Spanning Trees

- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

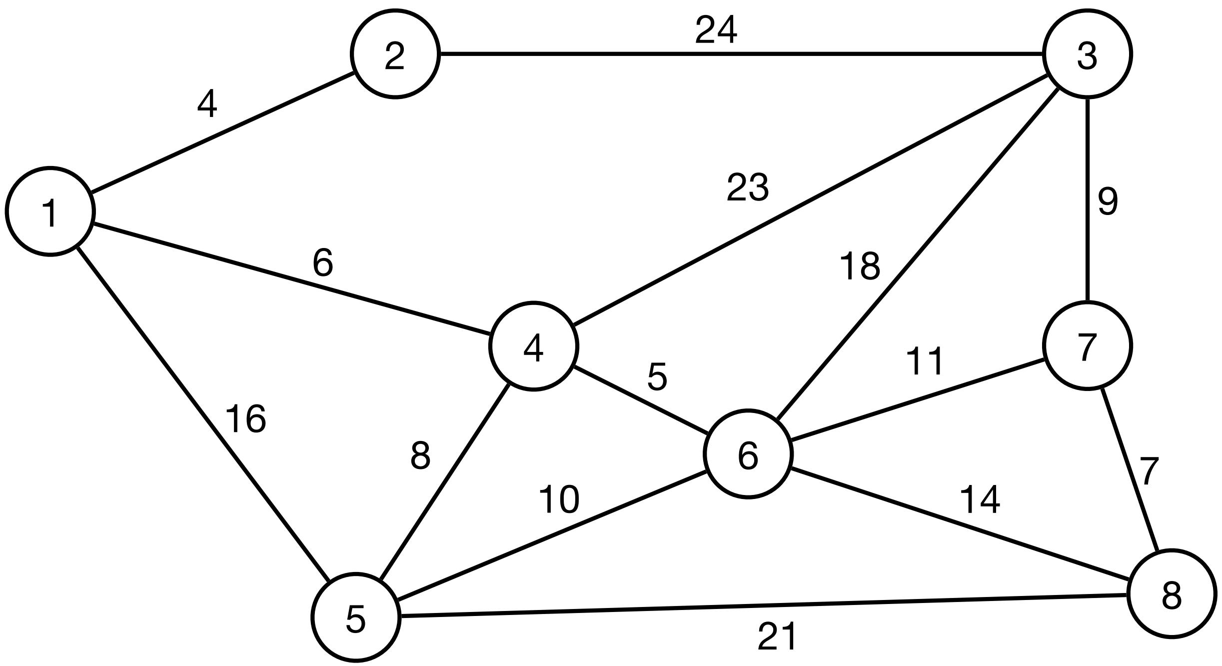
Philip Bille

Minimum Spanning Trees

- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

Minimum Spanning Trees

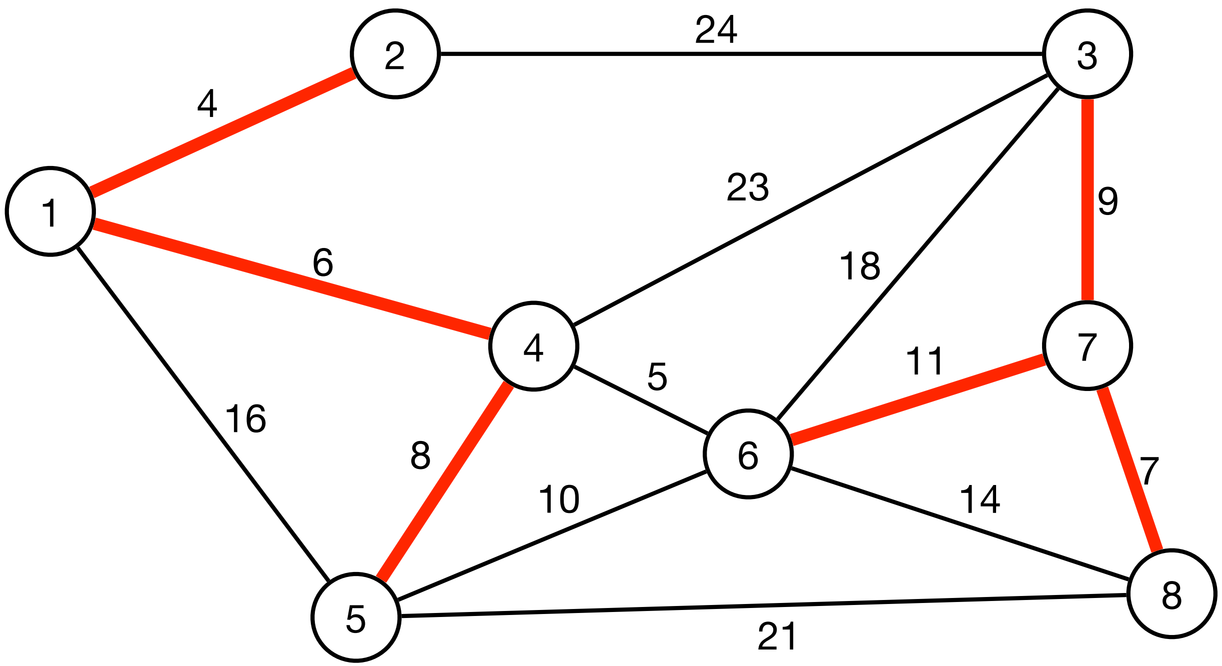
- Weighted graphs. Weight w(e) on each edge e in G.
- Spanning tree. Subgraph T of G over all vertices that is connected and acyclic.
- Minimum spanning tree (MST). Spanning tree of minimum total weight.



Graph G

Minimum Spanning Trees

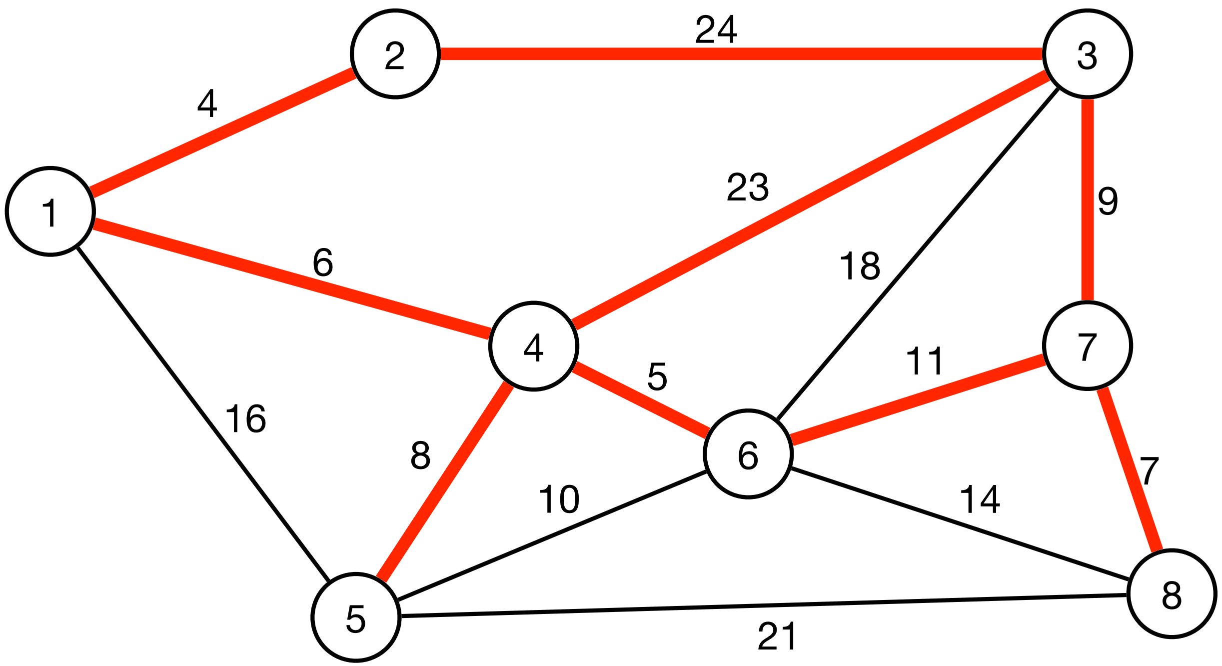
- Weighted graphs. Weight w(e) on each edge e in G.
- Spanning tree. Subgraph T of G over all vertices that is connected and acyclic.
- Minimum spanning tree (MST). Spanning tree of minimum total weight.



Not connected

Minimum Spanning Trees

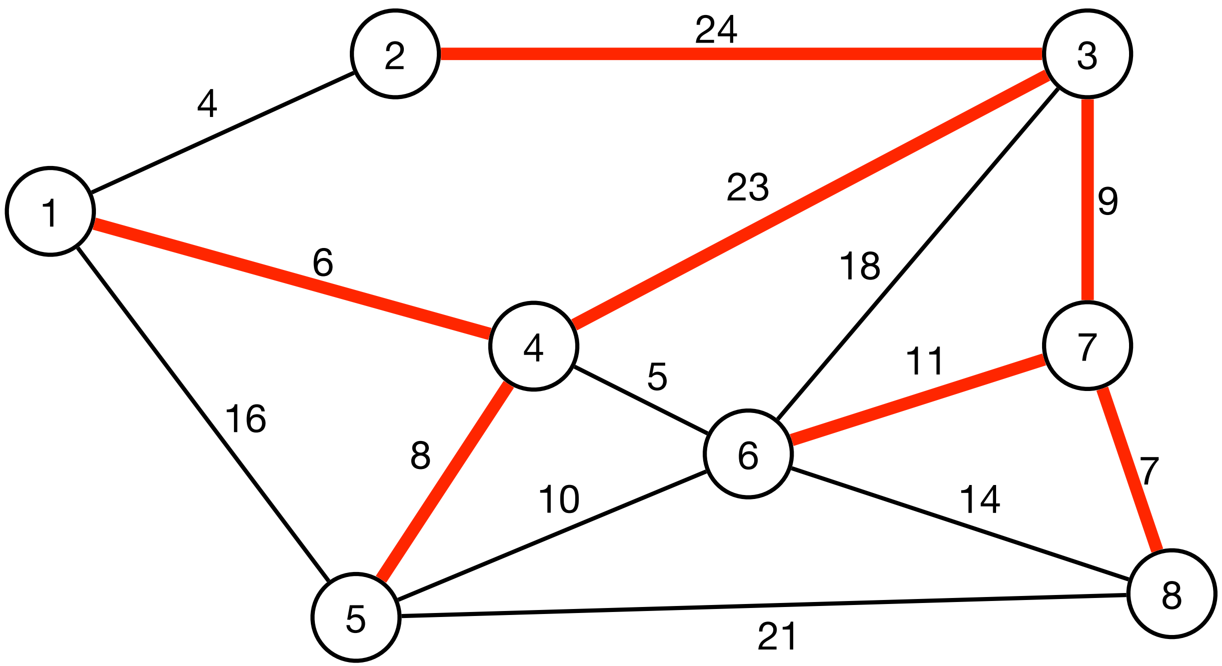
- Weighted graphs. Weight w(e) on each edge e in G.
- Spanning tree. Subgraph T of G over all vertices that is connected and acyclic.
- Minimum spanning tree (MST). Spanning tree of minimum total weight.



Connected and cyclic

Minimum Spanning Trees

- Weighted graphs. Weight w(e) on each e in G.
- Spanning tree. Subgraph T of G over all vertices that is connected and acyclic.
- Minimum spanning tree (MST). Spanning tree of minimum total weight.

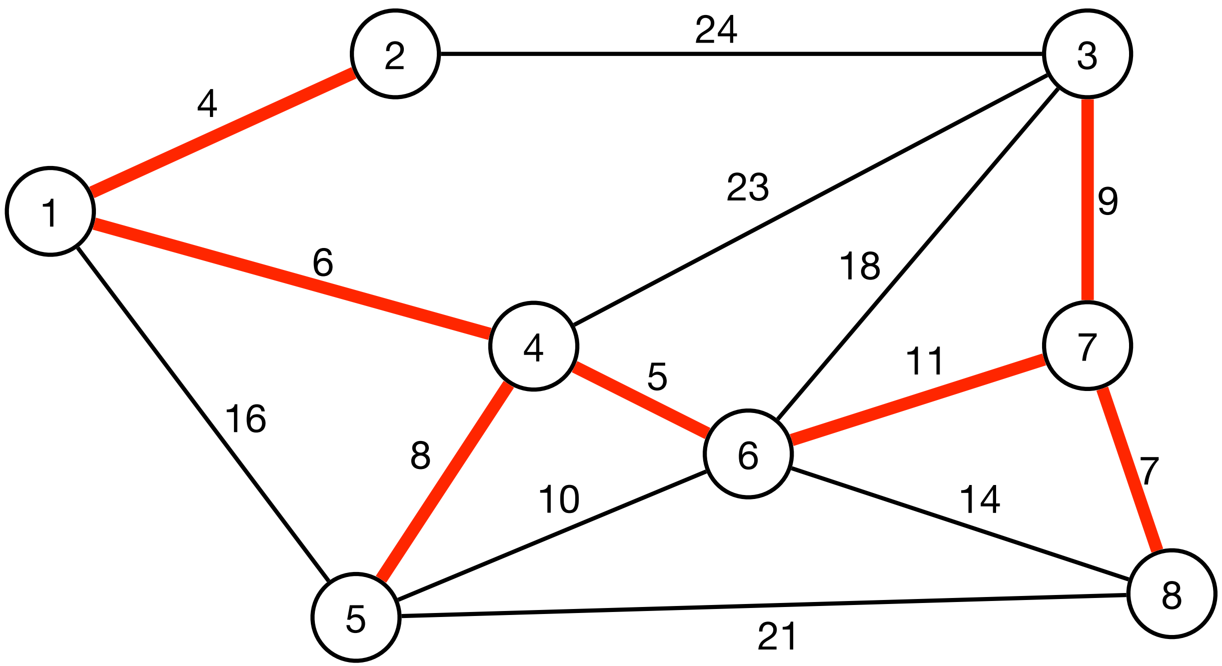


Connected and acyclic = spanning tree

Total weight = 6 + 8 + 23 + 24 + 9 + 11 + 7 = 88

Minimum Spanning Trees

- Weighted graphs. Weight w(e) on each e in G.
- Spanning tree. Subgraph T of G over all vertices that is connected and acyclic.
- Minimum spanning tree (MST). Spanning tree of minimum total weight.



Minimum spanning tree
Total weight = 4 + 6 + 5 + 8 + 11 + 9 + 7 = 50

Applications

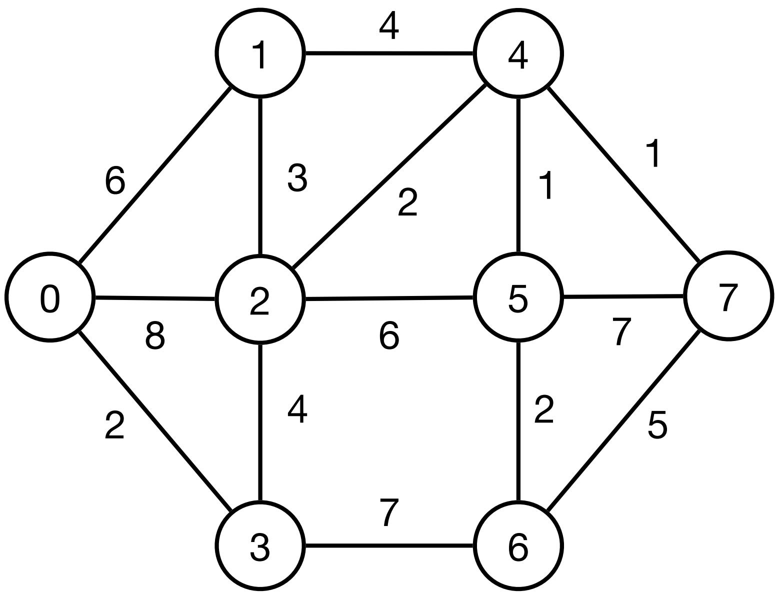
- Network design.
  - Computer, road, telephone, electrical, circuit, cable tv, hydralic, …
- Approximation algorithms.
  - Travelling salesperson problem, steiner trees.
- Other applications.
  - Meteorology, cosmology, biomedical analysis, encoding, image analysis, ...

Minimum Spanning Trees

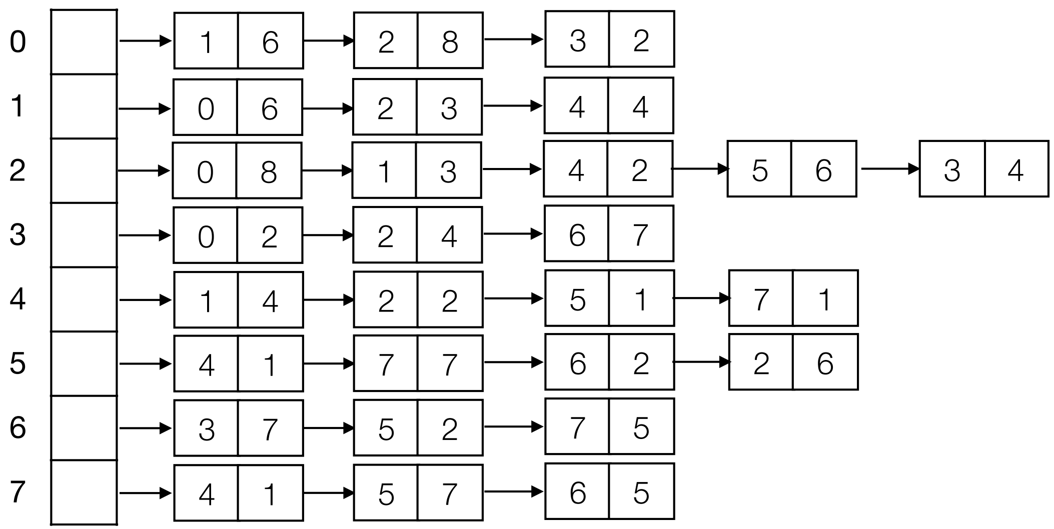
- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

Representation of Weighted Graphs

- Adjacency matrix and adjacency list.
- Similar for directed graphs.



	0	1	2	3	4	5	6	7
0	0	6	8	2	0	0	0	0
1	6	0	3	0	4	0	0	0
2	8	3	0	4	2	6	0	0
3	2	0	4	0	0	0	7	0
4	0	4	2	0	0	1	0	1
5	0	0	6	0	1	0	2	7
6	0	0	0	7	0	2	0	5
7	0	0	0	0	1	7	5	0



Minimum Spanning Trees

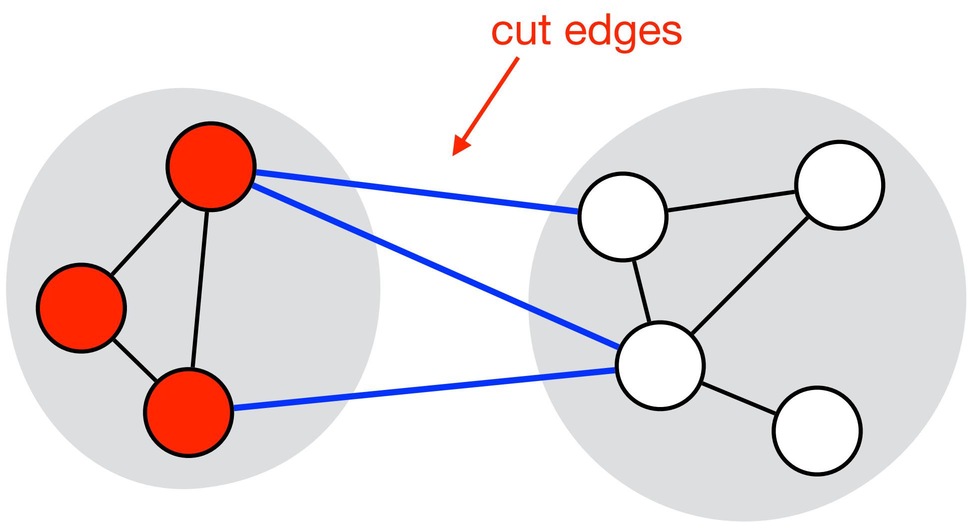
- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

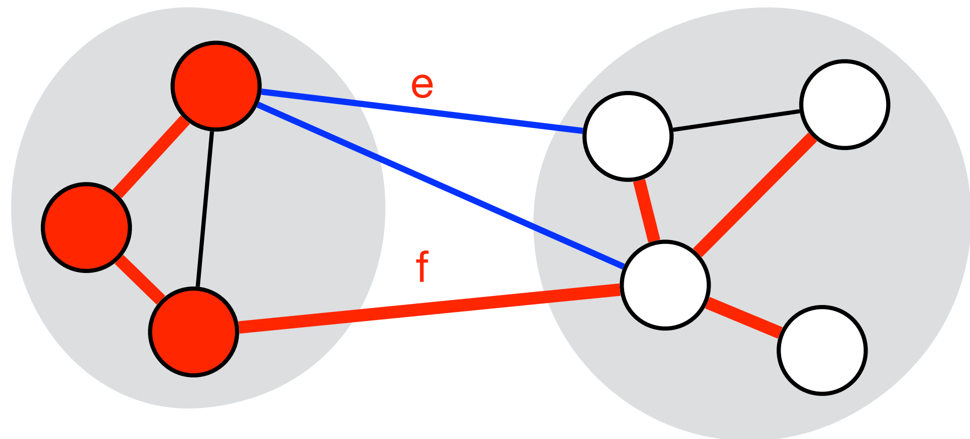
Properties of Minimum Spanning Trees

- Assume for simplicity:
  - All edge weights are distinct.
  - G is connected.
- $\Longrightarrow$ MST exists and is unique.

Cut Property

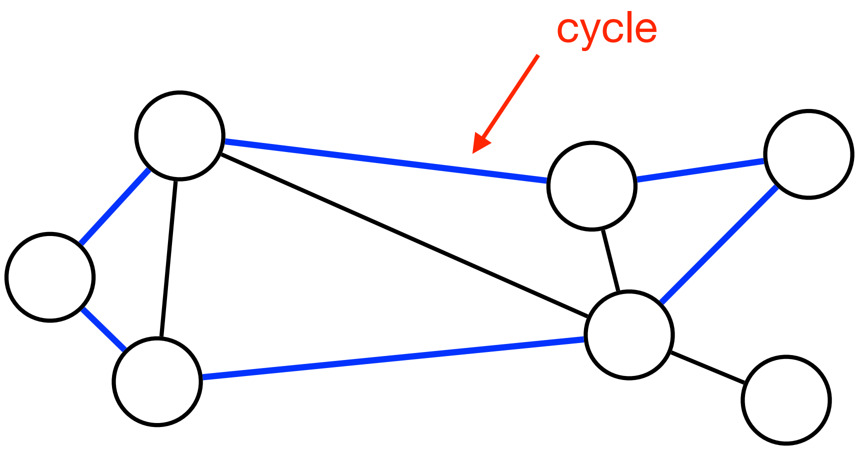
- Def. A cut is a partition of the vertices into two non-empty sets.
- Def. A cut edge is an edge crossing the cut.
- Cut property. For any cut, the lightest cut edge is in the MST.
- Proof.
    - Assume the lightest cut edge e is not in the MST.
    - Replace other cut edge f with e.
    - Produces a new spanning with smaller weight.

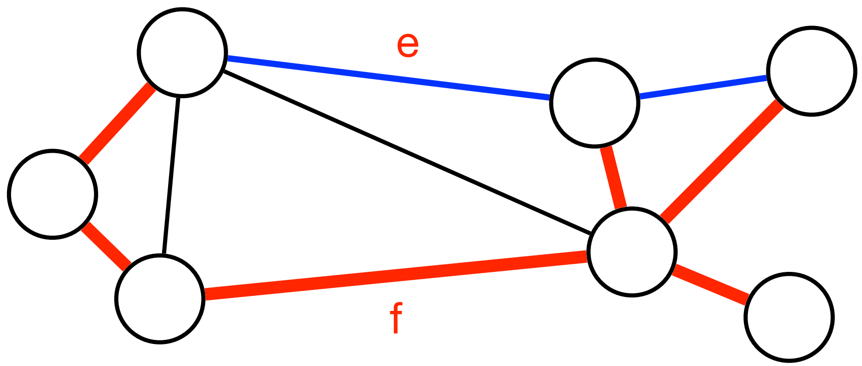




Cycle Property

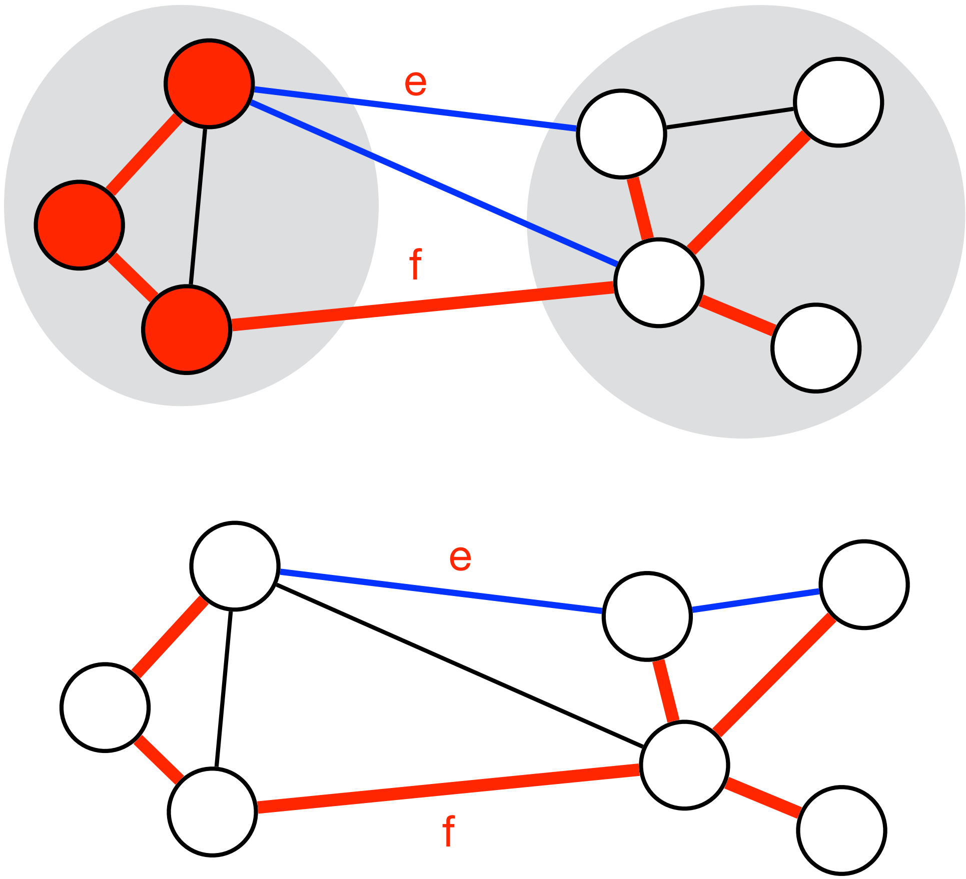
- Cycle property. For any cycle, the heaviest edge is not in the MST.
- Proof.
  - Assume heaviest edge f in cycle is in MST.
  - Replace f with lighter edge e in cycle.
  - Produces a new spanning tree with smaller weight.





Properties of Minimum Spanning Trees

- Cut property. For any cut, the lightest cut edge is in the MST.
- Cycle property. For any cycle, the heaviest edge is not in the MST.

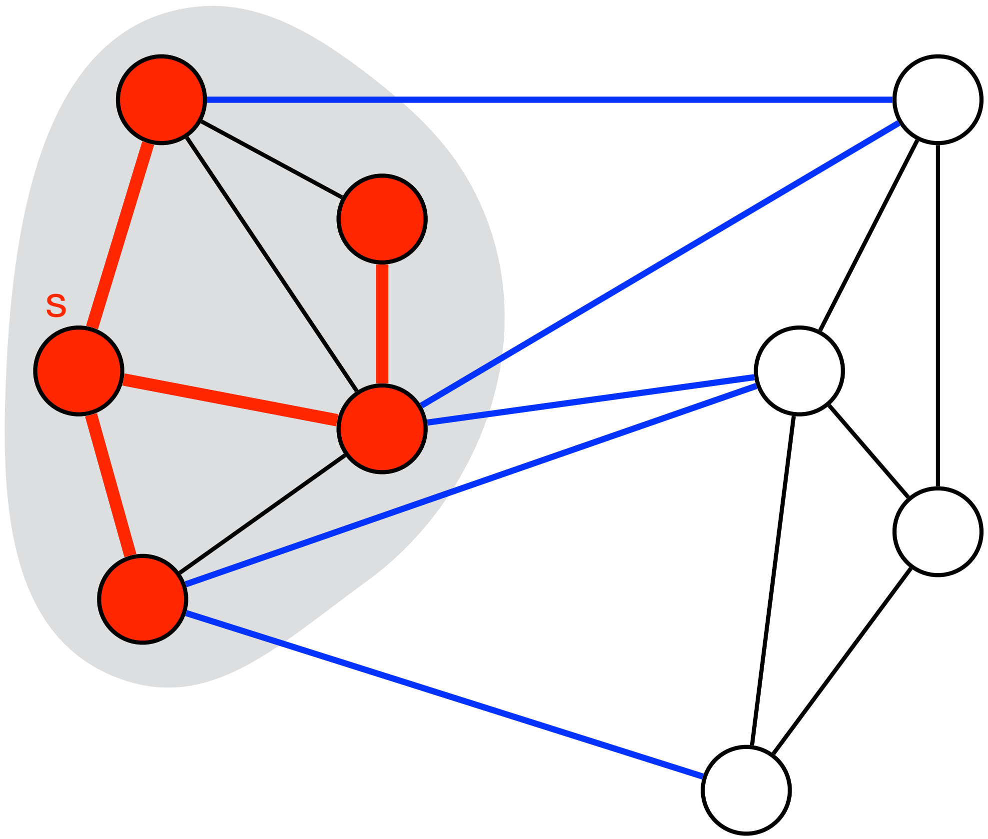


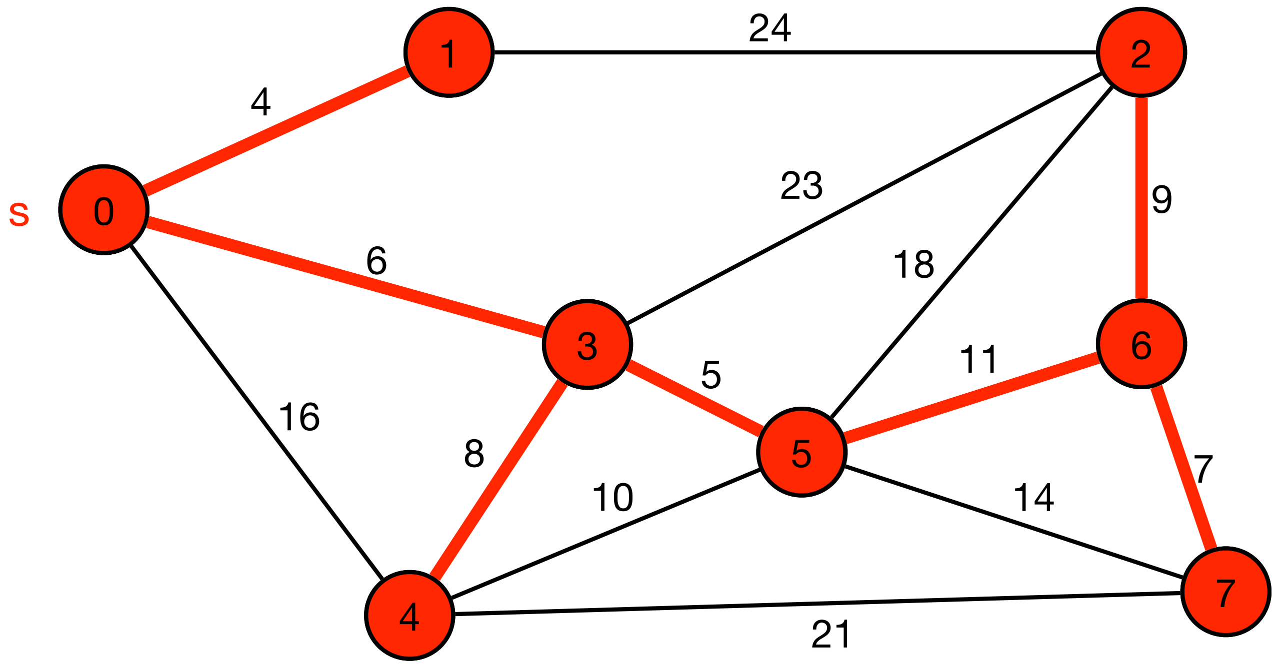
Minimum Spanning Trees

- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

Prim's Algorithm

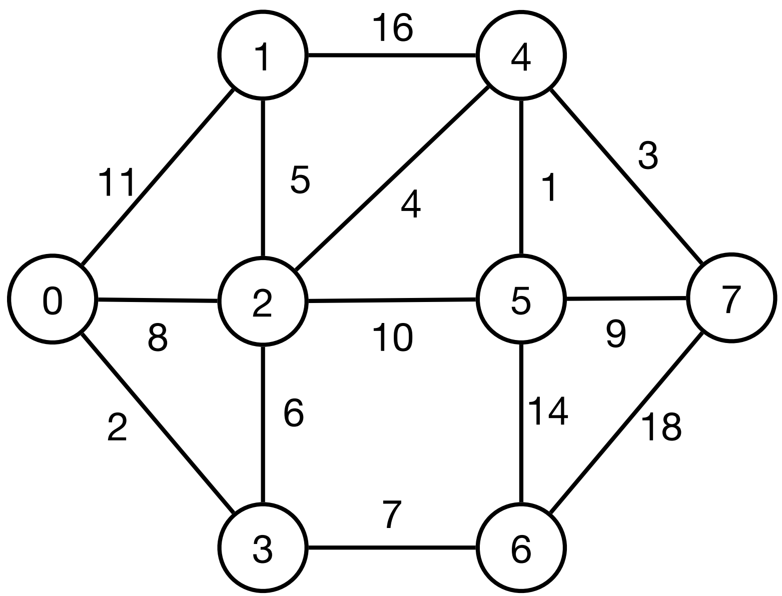
- Grow a tree T from some vertex s.
- In each step, add lightest edge with one endpoint i T.
- Stop when T has n-1 edges.




s
0
1
2
3
4
5
6
7
4
24
6
23
9
18
16
8
5
11
10
14
7
21

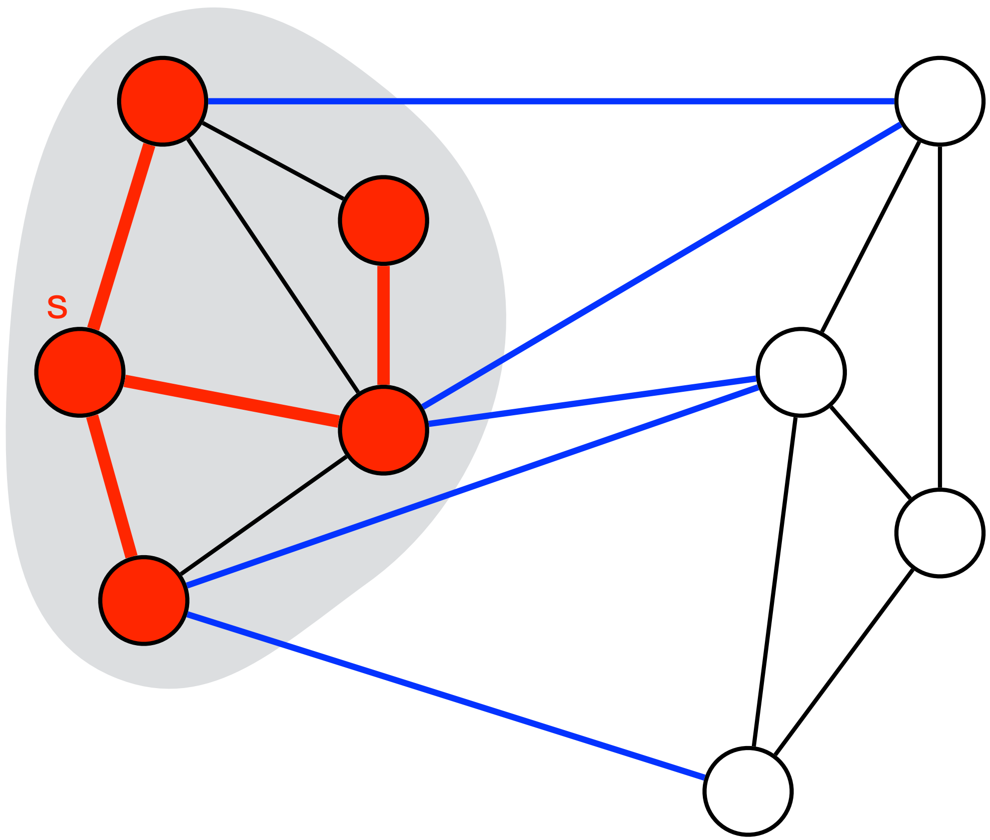
Prim's Algorithm

- Grow a tree T from some vertex s.
- In each step, add lightest edge with one endpoint i T.
- Stop when T has n-1 edges.
- Exercise. Show execution of Prim's algorithm from vertex 0 on the following graph.



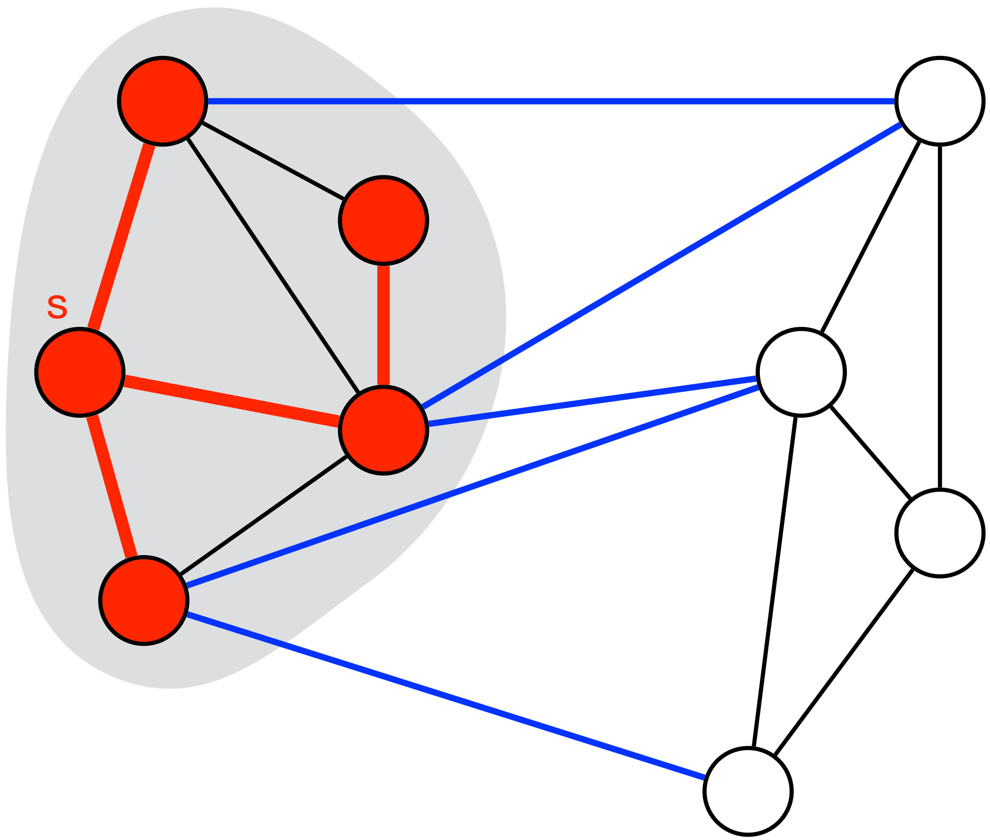
Prim's Algorithm

- Lemma. Prim's algorithm computes the MST.
- Proof.
  - Consider cut between T and other vertices.
  - We add lightest cut edge to T.
  - Cut property $\Longrightarrow$ edge is in MST $\Longrightarrow$ T is MST after n-1 steps.



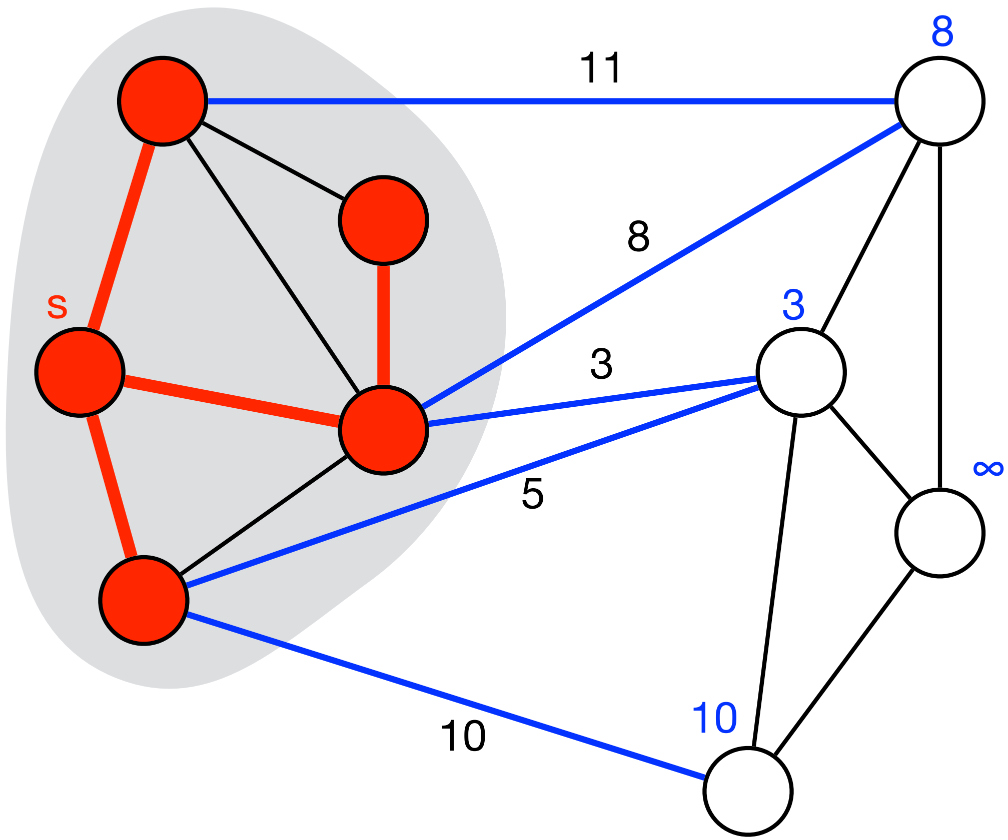
Prim's Algorithm

- Implementation. How do we implement Prim's algorithm?
- Challenge. Find the lightest cut edge.



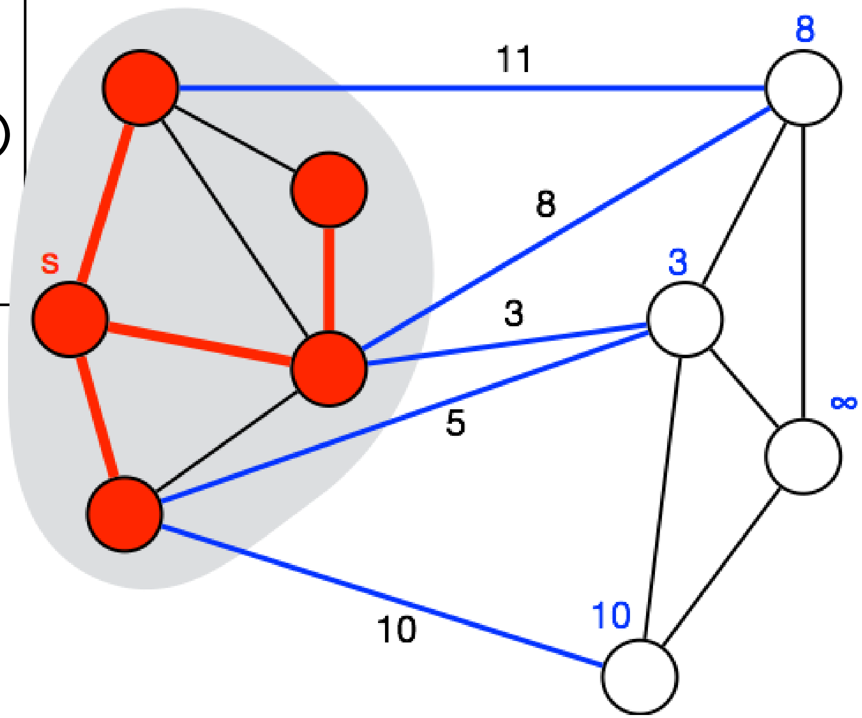
Prim's Algorithm

- Implementation. Maintain vertices outside T in priority queue.
  - Key of vertex v = weight of lightest cut edge (∞ if no cut edge).
  - In each step:
    - Find lightest edge = EXTRACT-MIN
    - Update weight of neighbors of new vertex with DECREASE-KEY.



Prim's Algorithm

```
PRIM(G, s)
    for all vertices v∈V
        v.key = ∞
        v.π = null
        INSERT(P,v)
    DECREASE-KEY(P,s,0)
    while (P ≠ ∅)
        u = EXTRACT-MIN(P)
        for all neighbors v of u
            if (v ∈ P and w(u,v)<key[v])
                DECREASE-KEY(P,v,w(u,v))
                v.π = u
```



- Time.
  - n EXTRACT-MIN
  - n INSERT
  - O(m) DECREASE-KEY
- Total time with min-heap. $O(n \log n + n \log n + m \log n) = O(m \log n)$

Prim's Algorithm

- Priority queues and Prim's algorithm. Complexity of Prim's algorithm depend on priority queue.
  - n INSERT
  - n EXTRACT-MIN
  - O(m) DECREASE-KEY

Priority queue	INSERT	EXTRACT-MIN	DECREASE-KEY	Total
array	O(1)	O(n)	O(1)	$O(n^2)$
binary heap	O(log n)	O(log n)	O(log n)	O(m log n)
Fibonacci heap	O(1)†	O(log n)†	O(1)†	O(m + n log n)

† = amortized

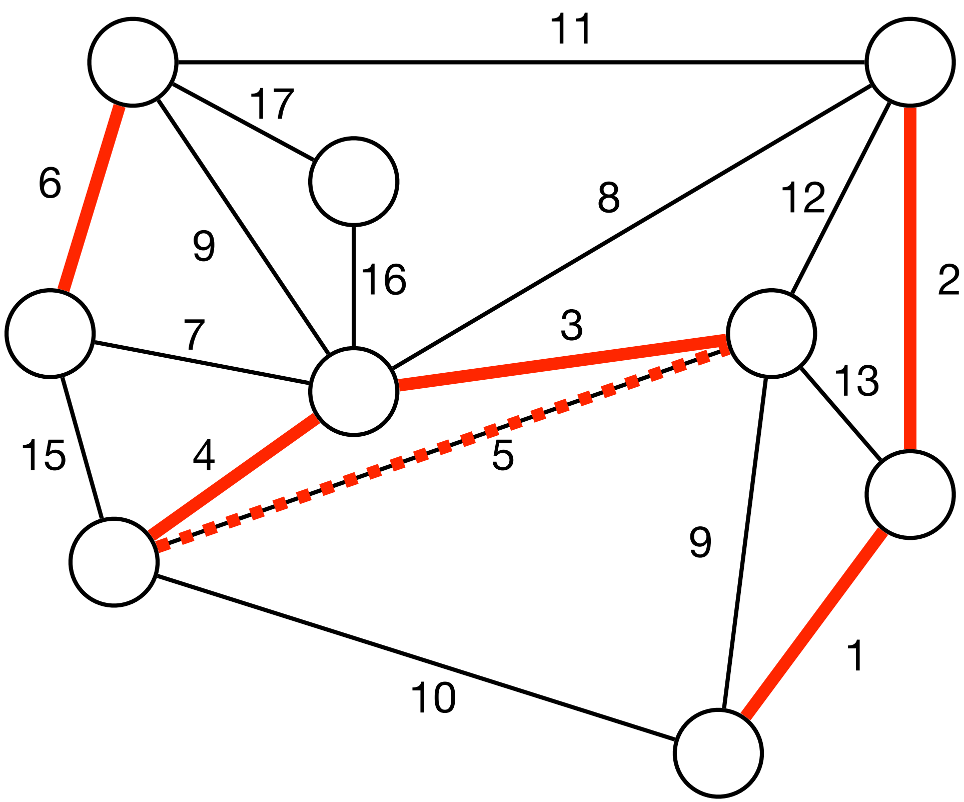
- Greed. Prim's algorithm is a greedy algorithm.
  - Makes local optimal choices in each step that lead to global optimal solution.

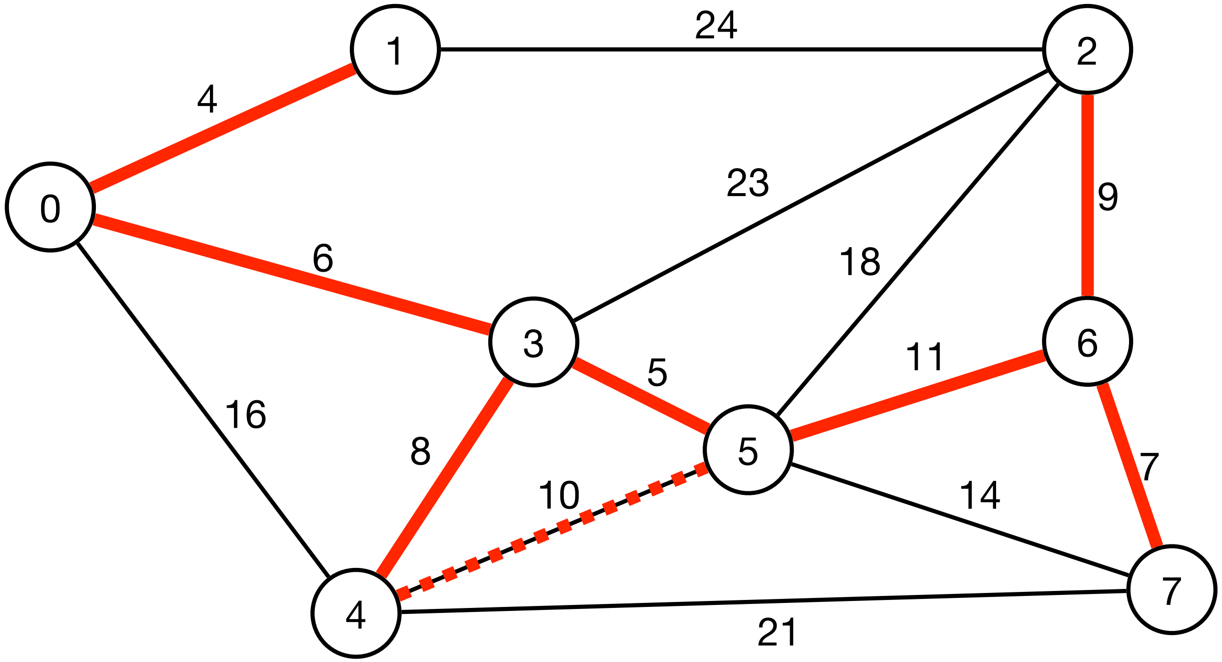
Minimum Spanning Trees

- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

Kruskal's Algorithm

- Consider edges from lightest to heaviest.
- In each step, add edge to T if it does not create a cycle.
- Stop when T has n-1 edges.

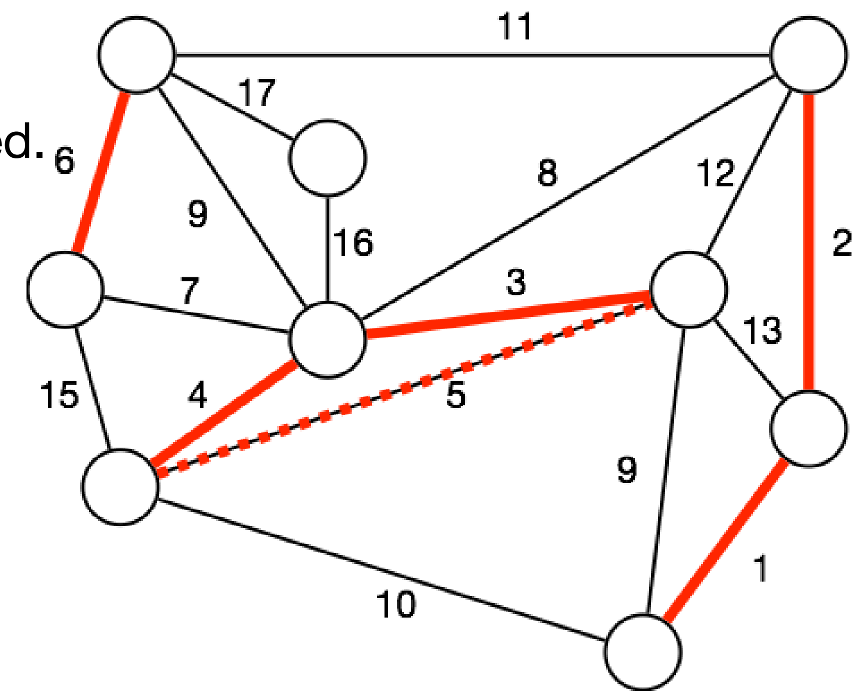




0
1
2
3
4
5
6
7
4
24
6
23
9
18
16
5
11
8
10
7
14
21

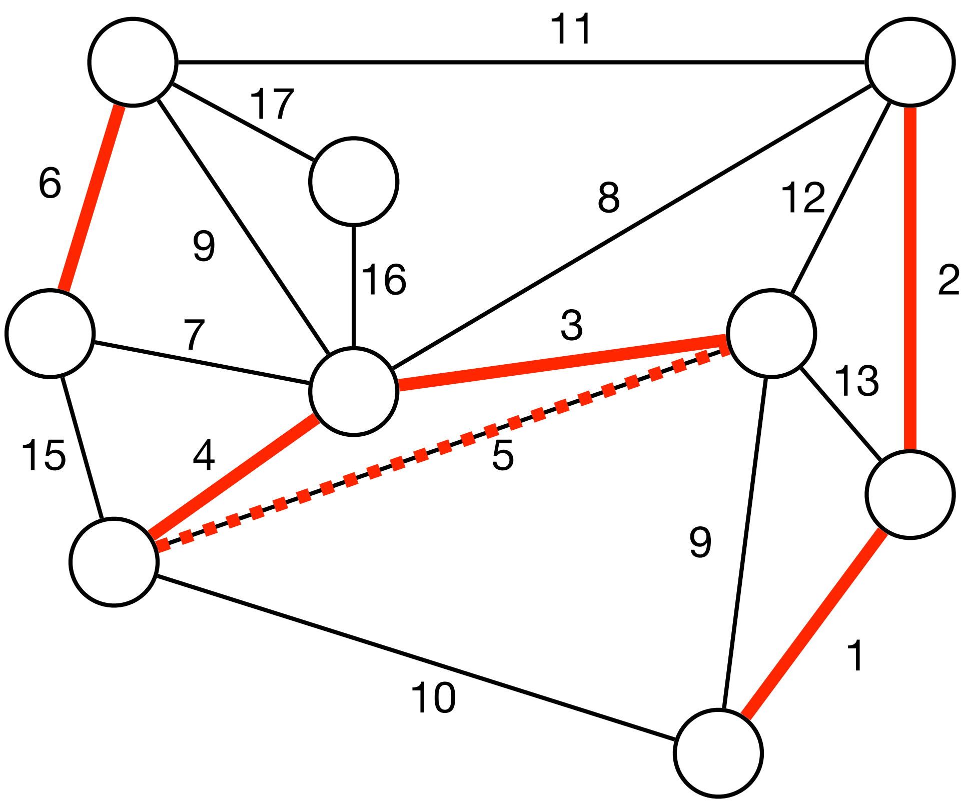
Kruskal's Algorithm

- Lemma. Kruskal's algorithm computes the MST.
- Proof.
  - Algorithms considers edges from light to heavy. At edge e = (u,v):
  - Case 1. e creates a cycle and is not added to T.
    - e must be heaviest edge on cycle.
    - Cycle property $\Longrightarrow$ e is not in MST.
  - Case 2. e does not create a cycle and is added to T.
    - e must be lightest edge in cut.
    - Cut property $\Longrightarrow$ e is in MST.
  - $\Longrightarrow$ T is MST when n-1 edges are added.



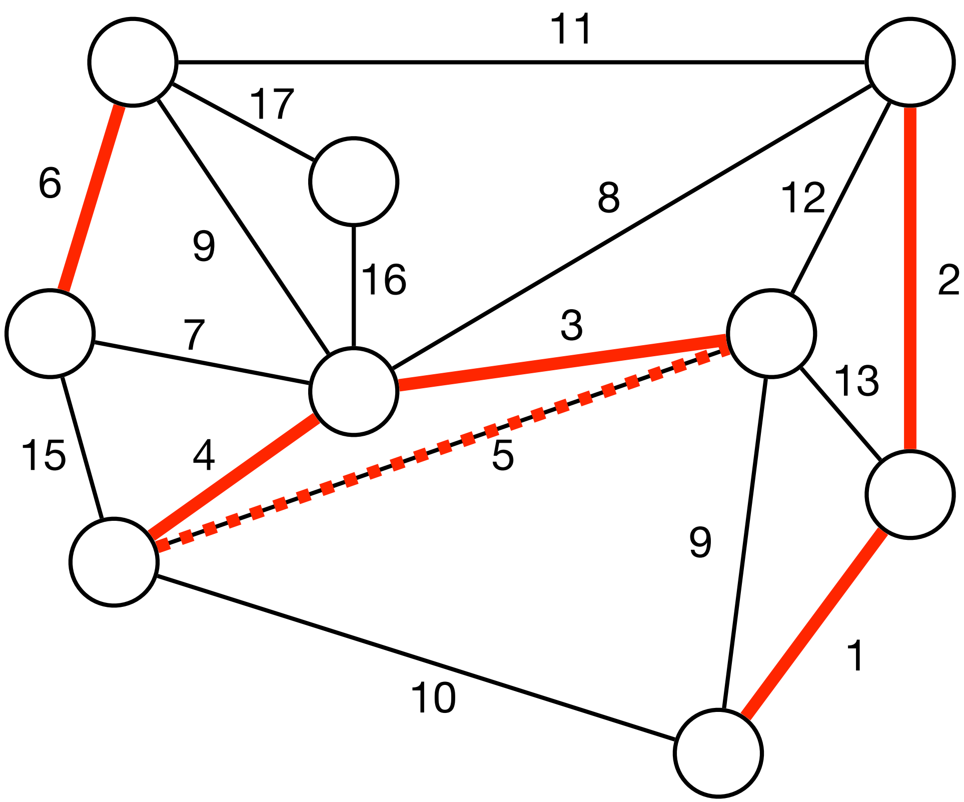
Kruskal's Algorithm

- Implementation. How do we implement Kruskal's algorithm?
- Challenge. Check if an edge form a cycle.



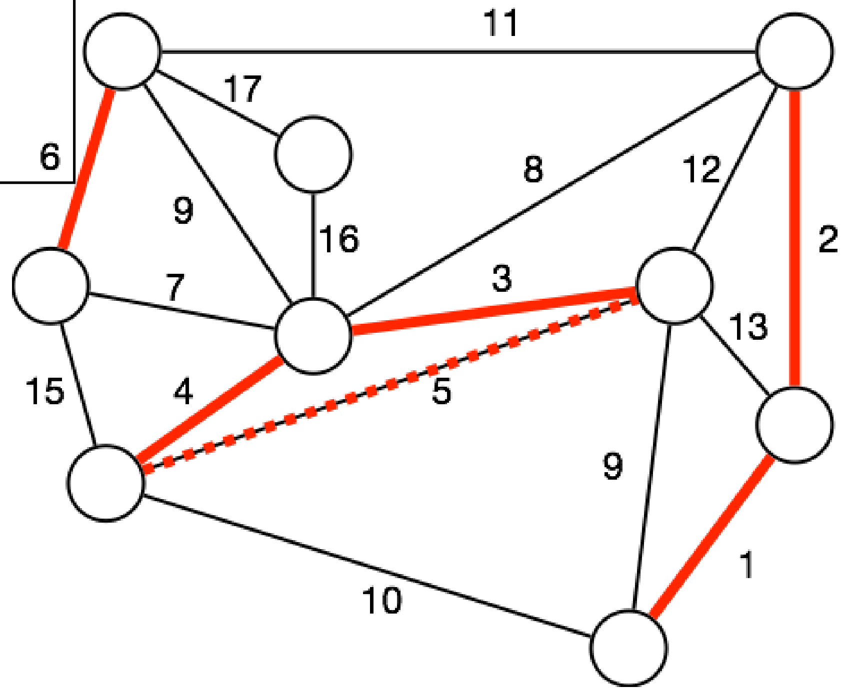
Kruskal's Algorithm

- Implementation. Maintain edges in a data structure for dynamic connectivity.
- In each step:
  - Check if an edge creates a cycle = CONNECTED.
  - Add new edge = INSERT.



Kruskal's Algorithm

```
KRUSKAL(G)
   Sort edges
   INIT(n)
   for all edges (u,v) i sorted order
      if (!CONNECTED(u,v))
         INSERT(u,v)
   return all inserted edges
```



- Time.
  - Sorting m edges.
  - 1 INIT
  - m CONNECTED
  - n INSERT
- Total time. O(m log m + n + m log n + n log n) = O(m log n).
- Greed. Kruskal's algorithm is also a greedy algorithm.

Minimum Spanning Trees

- What is the best algorithm for computing MSTs?

Year	Time	Authors
???	O(m log n)	Jarnik, Prim, Dijkstra, Kruskal, Boruvka, ?
1975	O(m log log n)	Yao
1986	$O(m \log^* n)$	Fredman, Tarjan
1995	O(m)‡	Karger, Klein, Tarjan
2000	$O(n\alpha(m,n))$	Chazelle
2002	optimal	Pettie, Ramachandran

‡ = randomized

Minimum Spanning Trees

- Minimum Spanning Trees
- Representation of Weighted Graphs
- Properties of Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm